# Classes and Objects

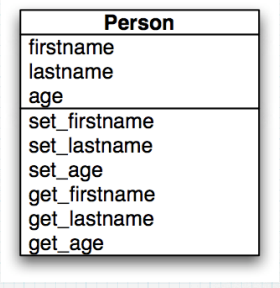
PHP

## Classes / Objects

- Combine values and process in a single data structure
- Closer to real world structures
- Design in UML class diagrams

## Person class

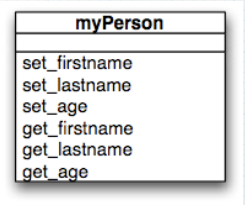
- Three properties
- Six methods



## Using the Person class

To use the class, create an Object of that class using the new keyword

```
$myPerson = new Person();
```

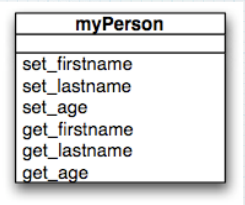



$myPerson is an Object of type Person

## Using the Person class

PHP uses -> to access the properties and methods inside the object

```
$myPerson->set_firstname('Homer');
$myPerson->set_lastname('Simpson');
$myPerson->set_age(38);
```

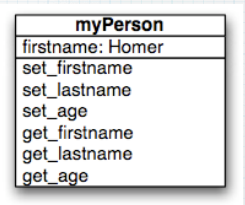



The set_ methods pass values into the object properties

## Using the Person class

PHP uses -> to access the properties and methods inside the object

```
$myPerson->set_firstname('Homer');
$myPerson->set_lastname('Simpson');
$myPerson->set_age(38);
```

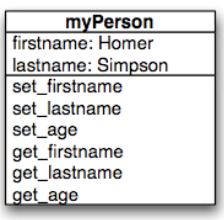



The set_ methods pass values into the object properties

## Using the Person class

PHP uses -> to access the properties and methods inside the object

```
$myPerson->set_firstname('Homer');
$myPerson->set_lastname('Simpson');
$myPerson->set_age(38);
```

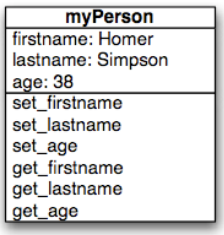



The set_ methods pass values into the object properties

## Using the Person class

PHP uses -> to access the properties and methods inside the object

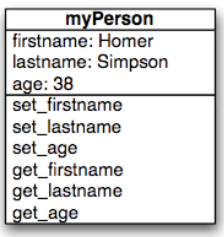
```
$myPerson->set_firstname('Homer');
$myPerson->set_lastname('Simpson');
$myPerson->set_age(38);
```




The set_ methods pass values into the object properties

## Using the Person class

PHP uses -> to access the properties and methods inside the object




```
print $myPerson->get_firstname();    Homer
print $myPerson->get_lastname();     Simpson
print $myPerson->get_age();          38
```

The get_ methods get values out of the object properties

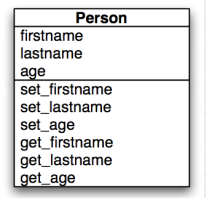
## Class syntax

```
class classname [ extends baseclass ]
{
    [ var $property [= value ]; ... ]
    [ function functionname (args) {
          // code
      }
      ...
    ]
}
```

## PHP implementation

```
class Person {
   var $firstname;
   var $lastname;
   var $age;

   ...

};
```

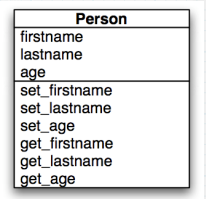
Three properties in the Person class



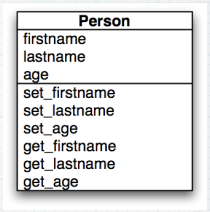

## PHP implementation

```
...

  function get_firstname(){
  return $this->firstname;
  }

  function get_lastname(){
  return $this->lastname;
  }

  function get_age(){
  return $this->age;
  }
...
```

Three methods to get the values out of the object



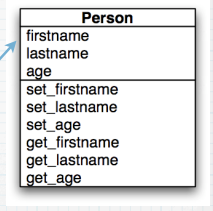
## PHP implementation




```
...
  function get_firstname(){
  return $this->firstname;
  }

  function get_lastname(){
  return $this->lastname;
  }

  function get_age(){
  return $this->age;
  }
...
```

Three methods to get the values out of the object

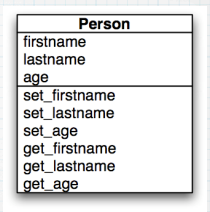
## PHP implementation




```
function get_firstname(){
return $this->firstname;
}
```

The $this keyword points to the containing object

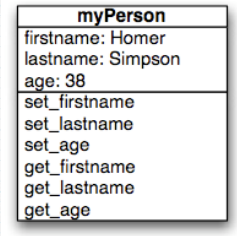
## PHP implementation




```
function get_firstname(){
return $this->firstname;
}
```

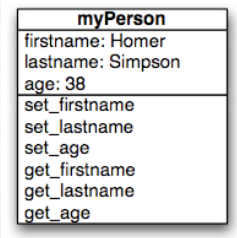
Note no $ used here

## PHP implementation



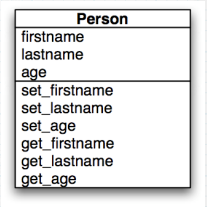

```
function get_firstname(){
return $this->firstname;
}
```

## PHP implementation




```
function get_firstname(){
return 'Homer';
}
```

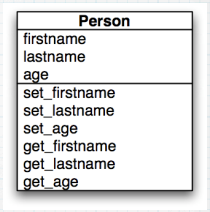
## PHP implementation




```
...
   function set_firstname($new_name){
   $this->firstname=$new_name;
   }

   function set_lastname($new_name){
   $this->lastname=$new_name;
   }

   function set_age($new_age){
   $this->age=$new_age;
   }
};
```

Three methods to set the values inside the object

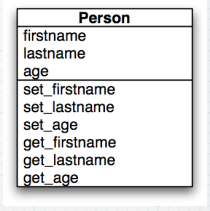
## PHP implementation




```
...
  function set_firstname($new_name){
  $this->firstname=$new_name;
  }

  function set_lastname($new_name){
  $this->lastname=$new_name;
  }

  function set_age($new_age){
  $this->age=$new_age;
  }
};
```

Three methods to set the values inside the object

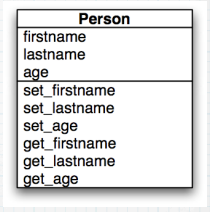
## PHP implementation




```
$myPerson->set_firstname('Homer');
```

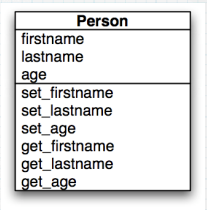
Call the set_firstname method inside myPerson

## PHP implementation




```
  function set_firstname($new_name){
  $this->firstname=$new_name;
  }
```

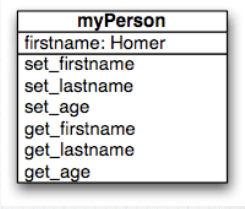
## PHP implementation




```
function set_firstname('Homer'){
$this->firstname=$new_name;
}
```

Value 'Homer' is passed into the function

## PHP implementation

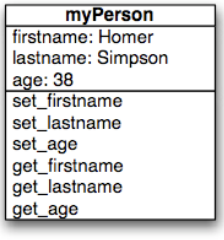



```
function set_firstname('Homer'){
$this->firstname='Homer';
}
```

Attribute firstname takes the value 'Homer'

## Encapsulation

You should always access the properties through the methods of the object




```
$myPerson->age=42;
```

Don't do this

```
$myPerson->set_age(42);
```

Do this

## Demo: Person example

```
<?
require('class.Person.php');
$myPerson = new Person();
$myPerson->set_firstname('Homer');
$myPerson->set_lastname('Simpson');
$myPerson->set_age(38);
?>

<html>
<head>
<title>Person Demo</title>
</head>
<body>
<h1>Person Demo</h1>
<p>My name is <? print $myPerson->get_firstname().' '.
    $myPerson->get_lastname(); ?></p>
<p>I am <? print $myPerson->get_age(); ?></p>


</body>
</html>
```

## Using a Constructor (PHP 5)

Rather than create the object and then pass in the initial values...

```
$myPerson = new Person();
$myPerson->set_firstname('Homer');
$myPerson->set_lastname('Simpson');
$myPerson->set_age(38);
```

Use a constructor function

## Using a Constructor (PHP 4)

The constructor function passes values in as we create the object:

```
$myPerson = new Person("Homer","Simpson",38);
```

## Using a Constructor (PHP 5)

### The constructor function is declared inside the object:

```
class Person {
   var $firstname;
   var $lastname;
   var $age;

   function __construct($newfirstname, $newlastname, $newage){
      $this->firstname=$newfirstname;
      $this->lastname=$newlastname;
      $this->age=$newage;
   }
```

## Demo: Person constructor example

```
<?
require('class.Person1.php');
$myPerson = new Person('Homer','Simpson',38);
?>

<html>
<head>
<title>Person Demo</title>
</head>
<body>
<h1>Person Demo</h1>
<p>My name is <? print $myPerson->get_firstname().' '.
$myPerson->get_lastname(); ?></p>
<p>I am <? print $myPerson->get_age(); ?></p>


</body>
</html>
```

## Default values

### You can set default values if none is provided

```
class Person {
   var $firstname;
   var $lastname;
   var $age;

   function __construct($newfirstname, $newlastname, $newage){
      $this->firstname=$newfirstname;
      $this->lastname=$newlastname;
      $this->age=$newage;
   }
```

## Default values

### You can set default values if none is provided

```
class Person {
   var $firstname;
   var $lastname;
   var $age;

   function __construct($newfirstname="", $newlastname="", $newage=0){
      $this->firstname=$newfirstname;
      $this->lastname=$newlastname;
      $this->age=$newage;
   }
```

## Demo: Person default values example

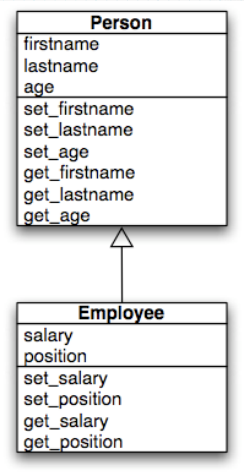
```
<?
require('class.Person1.php');
$myPerson = new Person('','Simpson');
?>

<html>
<head>
<title>Person Demo</title>
</head>
<body>
<h1>Person Demo</h1>
<p>My name is <? print $myPerson->get_firstname().' '.
$myPerson->get_lastname(); ?></p>
<p>I am <? print $myPerson->get_age(); ?></p>


</body>
</html>
```

## Inheritance

- PHP allows single inheritance for specialisation (or generalization)
- Properties and Methods may be added to child classes

## Inheritance

Person is the parent




Employee is the descendant of Person and inherits all the properties and methods

## Inheritance

Uses the extends keyword:

```
require_once("class.Person1.php");

class Employee extends Person {
   var $salary;
   var $position;
```

## Inheritance

PHP doesn't offer automatic chaining of constructors like some languages - so do it by hand:

```
class Employee extends Person {
   var $salary;
   var $position;

   function __construct($newfirstname, $newlastname,
     $newage,$newsalary,$newposition ){
      parent::_construct($newfirstname, $newlastname,
   $newage);
      $this->position=$newposition;
      $this->salary=$newsalary;
   }
```

## Inheritance

### The rest of Employee consists of extra get / set methods

```
function get_salary(){
return $this->salary;
}

function get_position(){
return $this->position;
}

function set_salary($new_salary){
$this->salary=$new_salary;
}

function set_position($new_position){
$this->position=$new_position;
}
```

## Demo: Employee example

```
<?
require_once('class.Employee.php');
$myEmployee = new Employee("Homer","Simpson",42,42000,"Safety Manager");
?>

<html>
<head>
<title>Employee Demo</title>
</head>
<body>
<h1>Employee Demo</h1>
<p>My name is
<? print $myEmployee->get_firstname().' '.$myEmployee->get_lastname(); ?>
</p>
<p>I am
<? print $myEmployee->get_age(); ?>
</p>
<p>I earn <? print $myEmployee->get_salary(); ?> in my job as
<? print $myEmployee->get_position(); ?></p>


</body>
</html>
```

## Overriding a method

- If a descendant uses a method with the same name as the parent, the method overrides the parent methods

## Overriding a method

### happy_birthday method in Person:

```
function happy_birthday(){
   $this->age++;
   return ("Age is now: ".$this->age);
}
```

## Demo: Happy Birthday example

```
<?
require('class.Person2.php');
$myPerson = new Person('Homer','Simpson',38);
?>

<html>
<head>
<title>Person Demo</title>
</head>
<body>
<h1>Person Demo</h1>
<p>My name is <? print $myPerson->get_firstname().' '.
$myPerson->get_lastname(); ?></p>
<p>I am <? print $myPerson->get_age(); ?></p>
<p>Happy birthday to me!
<? print $myPerson->happy_birthday(); ?></p>

</body>
</html>
```

## Overriding a method

### happy_birthday method in Employee:

```
function happy_birthday(){
return ("Employees have to buy cake: - Age is now ".
   $this->age);
   }
```

## Demo: Happy Birthday example for an Employee

```
<?
require_once('class.Employee2.php');
$myEmployee = new Employee("Homer","Simpson",
    42,"42000","Safety Manager");
?>

<html>
<head>
<title>Employee Demo</title>
</head>
<body>
<h1>Employee Demo</h1>
<p>My name is <? print $myEmployee->get_firstname().' '.
    $myEmployee->get_lastname(); ?></p>
<p>I am <? print $myEmployee->get_age(); ?></p>
<p>I earn <? print $myEmployee->get_salary(); ?> in my job as
<? print $myEmployee->get_position(); ?></p>
<p>Happy birthday to me!
<? print $myEmployee->happy_birthday(); ?></p>

</body>
</html>
```

## Overriding a method

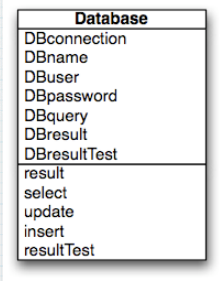
We may want the employee happy_birthday method to call the parent:

```
function happy_birthday(){
   parent::happy_birthday();
   return ("Employees have to buy cake: - Age is now ".
      $this->age);
}
```

## Demo: Happy Birthday example for an Employee

```
<?
require_once('class.Employee3.php');
$myEmployee = new Employee("Homer","Simpson",
    42,"42000","Safety Manager");
?>

<html>
<head>
<title>Employee Demo</title>
</head>
<body>
<h1>Employee Demo</h1>
<p>My name is <? print $myEmployee->get_firstname().' '.
    $myEmployee->get_lastname(); ?></p>
<p>I am <? print $myEmployee->get_age(); ?></p>
<p>I earn <? print $myEmployee->get_salary(); ?> in my job as
<? print $myEmployee->get_position(); ?></p>
<p>Happy birthday to me!
<? print $myEmployee->happy_birthday(); ?></p>

</body>
</html>
```

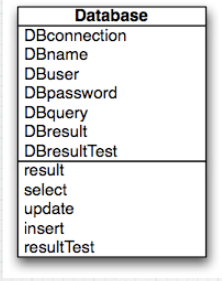
## Class example - Database




DBconnection, DBname, DBuser and DBpassword are used in the constructor to open a connection to the mySQL server

```
<?
require_once("../resources/class.Database.php");

$databaseConnection = new Database("localhost:8889","empdb","root","root");
```

## Class example - Database



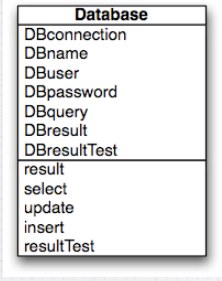

Methods select, update, insert are used to construct the SQL statement in DBquery

```
$databaseConnection->select("*","emp","sal>23000","","");
```

Select * from emp where sal>23000

Equivalent methods for update and insert

## Class example - Database




Use the result methods to return the result of running the query

```
$result=$databaseConnection->result();
```

Note that the current result is stored inside the object in DBresult

Extra method here - resultTest which can be used to see if any records were returned

# Example

## To open the empdb database and print the records

```
<?
require_once("../resources/class.Database.php");

$databaseConnection = new Database("localhost:8889","empdb","root","root");
$databaseConnection->select("*","emp","sal>23000","","");
$result=$databaseConnection->result();
?>

<html>
<head>
<title>Example SQL class</title>
</head>
<body>

<?
while ($row = mysql_fetch_array($result)){
   print $row["empno"]." ".$row["ename"]." ".$row["sal"]."<br />";
};
?>

</body>
</html>
```

Demo

# Class example - HMTL

| HTML |
| --- |
| HTMLdocument<br>title<br>stylesheet<br>head<br>body<br>bodycontent |
| append<br>buildHead<br>buildBody<br>addBodyContent<br>printHTML |

## title and stylesheet are set in the constructor (in the title and stylesheet properties)

```
<?
require_once("../resources/class.HTML.php");

$document = new HTML("Example document","");
```

## Leave title and stylesheet as empty strings if none required

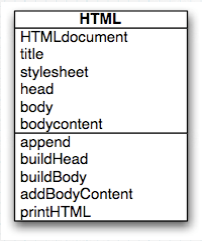
# Class example - HMTL

| HTML |
| --- |
| HTMLdocument<br>title<br>stylesheet<br>head<br>body<br>bodycontent |
| append<br>buildHead<br>buildBody<br>addBodyContent<br>printHTML |

## addBodyContent is used to add lines of HTML to the body (stored in bodycontent)

```
$document->addBodyContent("<h1>Heading example</h1>");
$document->addBodyContent("<p>Here is a paragraph</p>");
```

## Class example - HMTL




printHTML returns (as a string) the HTML which can be printed

```
print $document->printHTML();
```

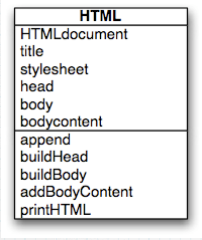
## Example

Creates a HTML object, adds some body content then prints it out

```
<?
require_once("../resources/class.HTML.php");

$document = new HTML("Example document","");
$document->addBodyContent("<h1>Heading example</h1>");
$document->addBodyContent("<p>Here is a paragraph</p>");
print $document->printHTML();
?>
```

Demo

## Class example - HMTL




append, buildHead, buildBody are used internally

printHTML assembles the HTML content using

buildHead which uses title and stylesheet

buildBody which uses body and bodycontent